Implementing a Unification Algorithm for Protocol Analysis with XOR

Max Tuengerthal[1], Ralf Küsters[1], and Mathieu Turuani[2]

[1] Christian-Albrechts-Universität zu Kiel, Germany
mtu@informatik.uni-kiel.de,kuesters@ti.informatik.uni-kiel.de
[2] Loria-INRIA, Vandoeuvre-lès-Nancy, France
mathieu.turuani@loria.fr

1 Introduction

Many methods and tools for the fully automatic analysis of security protocols are based on a technique called constraint solving (see, e.g., [11,7]), which as a central component involves a unification algorithm. The first methods and tools for the analysis of security protocols assumed the message space to be a free term algebra. However, this is a too idealized assumption in case the protocols employ operators involving algebraic properties, such as the exclusive or (XOR), an operator frequently used in security protocols. In [4,8] it was shown that the security, more precisely secrecy and authentication, of protocols is still decidable w.r.t. a bounded number of sessions, even NP-complete [4], when taking algebraic properties of XOR into account. However, these results do not yield practical algorithms. A first algorithm based on constraint solving and tailored towards efficient implementation was proposed by Chevalier [3]. However, a prerequisite for this algorithm to be of practical use is a unification algorithm for a combination of the equational theory E_{ACUN} (modeling algebraic properties of XOR) and an equational theory E_{std} modeling public/private keys which works well in practice. The goal of the present work is to provide such an algorithm.

A unification algorithm for $E = E_{\mathrm{std}} \cup E_{\mathrm{ACUN}}$ can easily be obtained by the general combination method proposed by Baader and Schulz [1], since unification algorithms for E_{std} and E_{ACUN} exist. However, this unification algorithm would be highly non-deterministic and therefore not directly suitable for practical use. Several optimizations have been proposed. First, Baader and Schulz [1] already suggested simple optimizations. More sophisticated optimizations, called iterative and deductive method, were presented by Kepser and Richts [10], who exploit concrete properties of the theories, like collapse-freeness, to limit the non-determinism. Another combination method, along with optimizations, was proposed by Boudet [2]. However, the settings in all of these works are still quite general and their optimizations do not suffice for our purposes.

In this paper, we propose a unification algorithm for the theory E which combines unification algorithms for E_{std} and E_{ACUN} but compared to the more general combination methods mentioned above uses specific properties of the equational theories for further optimizations. Our optimizations drastically reduce the number of non-deterministic choices, in particular those for variable identification and linear orderings. This is important for reducing both the runtime of the unification algorithm and the number of unifiers in the complete set of unifiers. We emphasize that

obtaining a "small" set of unifiers is essential for the efficiency of the constraint solving procedure within which the unification algorithm is used.

Outline of the Paper. In the following section, we briefly recall the combination algorithm by Baader and Schulz along with the optimizations proposed by Kepser and Richts. In Section 3, our unification algorithm is introduced, with experimental results presented in Section 4. We conclude in Section 5. Further details can be found in a technical report [13].

2 The General Combination Algorithm

In this section, we briefly describe the general combination method of Baader and Schulz [1] and optimizations introduced by Kepser and Richts [10] as our algorithm is based on [1] and some optimizations are motivated by [10].

Given disjoint equational theories E_1 and E_2 and stand-alone unification algorithms A_1 and A_2 for E_1 and E_2, respectively, which work with linear constant restrictions (see below) the combination method of Baader and Schulz combines A_1 and A_2 to obtain a unification algorithm for the joined theory $E = E_1 \cup E_2$. More precisely, given an elementary E-unification problem Γ, the combination method works as follows:

1. **Purification and splitting**. Obtain the sub-problems $\Gamma_{1,x}$, with $x \in \{1, 2\}$, by purifying terms and splitting equations for each theory E_x. (Non-pure terms or equations are those containing symbols of different theories.)
2. **Variable identification**. Choose a partition (i.e., equivalence classes) on variables for each $\Gamma_{1,x}$, $x \in \{1, 2\}$. Let $\Gamma_{2,x}$ be the sub-problem obtained from $\Gamma_{1,x}$ by replacing each variable by a representative of its class.
3. **Choose theory indices**. For each variable v in V choose a theory index $Ind(v) \in \{1, 2\}$ where V is the set of variables occurring in both $\Gamma_{2,1}$ and $\Gamma_{2,2}$. If in $\Gamma_{2,1}$ a variable has theory index 2 it is considered a constant in $\Gamma_{2,1}$; analogously for $\Gamma_{2,2}$.
4. **Choose linear ordering**. Choose a linear ordering $<$ on V. (Together with 3., the linear ordering $<$ induces what Baader and Schulz call a *linear constant restriction.*)
5. **Solve systems**. For each theory E_x, the algorithm A_x is applied to $\Gamma_{2,x}$ and $<$ to produce a complete set C_x of unifiers respecting $<$, where a unifier σ respects $<$ if $x < y$ implies that y does not occur in $x\sigma$ for every $x, y \in V$.
6. **Combine unifiers**. If C_1 or C_2 are not empty, combine the unifiers of C_1 with those of C_2 to obtain a set of E-unifiers of Γ. Go back to 2. to try other choices (in order to obtain further unifiers).

Theorem 1. *[1] The set of E-unifiers produced by the combination method above form a complete set of E-unifiers of the E-unification problem Γ.*

The major disadvantages of the general combination method are its high degree of non-determinism and the non-detection of failures before the last step. This results in poor runtime behavior and sets of unifiers that are far from minimal.

The main idea of the optimizations of Kepser and Richts [10] are to first make all non-deterministic decisions for one component in order to detect failures as soon as possible (iterative method) and to use constraints obtained by solving one component for reducing the number of remaining non-deterministic choices (deductive method).

3 Our Optimized Algorithm

We now present our unification algorithm for the equational theory $E = E_{\text{std}} \cup E_{\text{ACUN}}$ where $E_{\text{std}} = \{x \approx (x^{-1})^{-1}\}$ with $\cdot^{-1}$ modeling a mapping between public and private keys and $E_{\text{ACUN}} = \{x \oplus (y \oplus z) \approx (x \oplus y) \oplus z, x \oplus y \approx y \oplus x, x \oplus 0 \approx x, x \oplus x \approx 0\}$ for modeling the XOR operator. The theory E_{std} is associated with a signature containing finitely many free symbols of arbitrary arity, including constants or binary symbols for pairing $\langle \cdot, \cdot \rangle$ and encryption $\{\cdot\}$.. The signature associated with E_{ACUN} is $\{\oplus, 0\}$. We note that both E_{std} and E_{ACUN} are unitary for elementary unification and efficient unification algorithms exist for both theories. However, it is not hard to see that E is not unitary; by Theorem 1 E is finitary. Unification for E can easily be shown to be NP-complete using results in [9].

In what follows, we summarize the main optimizations of our algorithm compared to those discussed in the previous section, along with brief justifications of their correctness. Our optimizations employ specific properties of the equational theories under consideration and they reduce both the runtime and the size of complete unification sets.

Simplified iterative and deductive method. Similar to Kepser and Richts, we employ the idea of the iterative and deductive method but apply it only once to E_{std}. That is, we first solve the E_{std}-unification problem without any constraints. If this fails, the original problem is unsolvable. Otherwise, we obtain an mgu σ_{std} used in subsequent steps to reduce the number of non-deterministic choices. Since typically the E_{ACUN}-unification problem will not yield further constraints, we postpone solving this unification problem to a later point.

Hierarchy of variable identifications. A major new optimization in our algorithm is that we do not have to iterate over all possible variable identifications. If unification for both Γ_{std} and Γ_{ACUN} succeeds for some variable identifications p and p' where p is more general than p', then the combined unifier for p is more general than the one for p'. This can be shown using the following property of E_{ACUN}:

Lemma 1. *Every mgu of a E_{ACUN}-unification problem with linear constant restriction is also an mgu of this unification problem without restrictions.*

The above property on variable identifications allows us to traverse the tree of variable identifications in a breadth-first manner and skip all less general variable identifications once we succeed in solving the problem for a more general one.

Reduce number of choices of indices. Most theory indices can be determined from σ_{std}. If a variable is instantiated by a term with a collapse-free top-symbol, then this variable has to be a constant in Γ_{ACUN}. On the other hand, if x is not instantiated by σ_{std} and if there exists no variable y with $y\sigma_{\text{std}} = x^{-1}$, then it does not matter whether x is treated as a constant in Γ_{std} or not. In fact, a non-deterministic choice of theory indices must only be made for variables x and y such that $x\sigma_{\text{std}} = y^{-1}$ and $y\sigma_{\text{std}} = y$. Of course, not both can be constants in Γ_{std}, so it suffice to choose one of them.

Reduce number of choices of linear orderings. Instead of choosing an arbitrary linear ordering on V (see Section 2), we first deduce (deterministically) a partial ordering $<_{po}$ from σ_{std} such that $x <_{po} y$ iff y occurs in $x\sigma_{\text{std}}$. Now, the

Table 1. Runtimes and sizes of complete sets of unifiers: "size" denotes the size of the returned complete set of unifiers; "vi opt" stands for "variable identification optimization"; x, y, z, u, x_i are variables and a, b, c, d, e are constants. Runtime tests obtained on a 1.5 GHz Intel Pentium M processor.

		with vi opt		without vi opt	
no	unification problem	time (msecs)	size	time (msecs)	size
1	$\langle x, \langle \{x \oplus y\}_a, \{\{x \oplus y\}_a \oplus z\}_a \rangle \rangle \stackrel{?}{=}_E \langle \{b \oplus c\}_a, \langle \{\{b \oplus c\}_a \oplus d\}_a, \{\{\{b \oplus c\}_a \oplus d\}_a \oplus e\}_a \rangle \rangle$	3.3	1	> 30 min	
2	$z \stackrel{?}{=}_E \{\langle x, \langle y, x \oplus y \rangle \rangle\}_{(z \oplus u)^{-1}}$	0.1	1	3.3	15
3	$z \stackrel{?}{=}_E \{\langle x, \langle y, x \oplus y \rangle \rangle\}_{(z \oplus a)^{-1}}$	9.1	0	9.1	0
4	$0 \stackrel{?}{=}_E \langle x_1, y_1 \rangle \oplus \ldots \oplus \langle x_9, y_9 \rangle$	9.6 s	0	9.6 s	0
5	$0 \stackrel{?}{=}_E \langle x_1, y_1 \rangle \oplus \ldots \oplus \langle x_{10}, y_{10} \rangle$	239.7 s	945	70.7 s	6556

important observation is that by Lemma 1 once we have found a solution of the E_{ACUN}-unification problem w.r.t. a linear ordering $<$ which extends $<_{po}$, we do not need to try other linear orderings.

Theorem 2. *The algorithm described above returns a complete set of E-unifiers for a given E-unification problem.*

We note that the optimizations explained above are fairly independent of the theory E_{std}. Hence, E_{std} can easily be replaced by other theories.

4 Experimental Results

Table 1 summarizes some of our experimental results (see [13] for more). It contains runtimes and sizes of complete sets of unifiers both with the optimization for variable identification turned on and off. (The other optimizations are harder to turn on and off in our implementation, which is why these optimization are always turned on.) These results show that our unification algorithm runs efficiently on many benchmarks and that our optimizations indeed reduce both runtime and size of complete sets unifiers. In fact, the optimized version of our algorithm always returned minimal sets of unifiers. (However, we have no proof that this is always the case.)

Problem 1 in Table 1 is a unification problem that occurs in the analysis of the recursive authentication protocol [12]. Interestingly, while our algorithm quickly returns an mgu, the version of the algorithm with the optimization for variable optimization turned off does not come back with a solution within 30 minutes. The two versions of the algorithm also perform very differently on problem 2. There is no difference in problem 3 since this problem is not unifiable, and hence, the algorithm has to try all possible variable identifications. Problems 4 and 5 are only of theoretical interest, they typically do not occur in applications but illustrate the limitations of optimizations. Note that in problems of this form the size of a minimal complete set of unifiers may be exponential in the size of the problems.

5 Conclusion

Motivated by the analysis of security protocols, we have presented a unification algorithm for an equational theory including ACUN. Our algorithm contains several optimizations which make use of the specific properties of the equational theories at hand and performs well on practical examples, both in terms of its runtime and the size of the complete set of unifiers returned. As such, our algorithm is well-suited as a subprocedure in constraint solving algorithms for security protocol analysis with XOR.

One future direction is to incorporate other operators and their algebraic properties into our algorithm, including important operators such as Diffie-Hellman Exponentiation and RSA encryption. In [5,6], it was shown that fully automatic analysis of security protocols is also possible in presence of such operators.

References

1. F. Baader and K.U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *J. of Symbolic Computation*, 21:211–243, 1996.
2. A. Boudet. Combining unification algorithms. *J. of Symb. Comput.*, 16(6):597–626, 1993.
3. Y. Chevalier. A Simple Constraint Solving Procedure for Protocols with Exclusive Or. In *UNIF 2004*, 2004.
4. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. An NP Decision Procedure for Protocol Insecurity with XOR. In *LICS 2003*, pages 261–270. IEEE, Computer Society Press, 2003.
5. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the Security of Protocols with Diffie-Hellman Exponentiation and Products in Exponents. In *FSTTCS 2003*, volume 2914 of *LNCS*, pages 124–135. Springer, 2003.
6. Y. Chevalier, R. Küsters, M. Rusinowitch, and M. Turuani. Deciding the Security of Protocols with Commuting Public Key Encryption. *Electronic Notes in Theoretical Computer Science*, 125(1):55–66, 2005.
7. Y. Chevalier and L. Vigneron. A Tool for Lazy Verification of Security Protocols. In *ASE 2001*, pages 373–376. IEEE CS Press, 2001.
8. H. Comon-Lundh and V. Shmatikov. Intruder deductions, constraint solving and insecurity decision in presence of exclusive or. In *LICS 2003*, pages 271–280. IEEE, Computer Society Press, 2003.
9. Q. Guo, P. Narendran, and D.A. Wolfram. Unification and matching modulo nilpotence. In *CADE 1996*, pages 261–274, 1996.
10. S. Kepser and J. Richts. Optimisation techniques for combining constraint solvers. In *Frontiers of Combining Systems 2, Papers presented at FroCoS'98*, pages 193–210. Research Studies Press/Wiley, 1999.
11. J.K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. In *CCS 2001*, pages 166–175. ACM Press, 2001.
12. P.Y.A. Ryan and S.A. Schneider. An Attack on a Recursive Authentication Protocol. *Information Processing Letters*, 65(1):7–10, 1998.
13. M. Tuengerthal. Implementing a unification algorithm for protocol analysis with XOR. Technical Report 0609, Institut für Informatik, CAU Kiel, Germany, 2006. Available from http://www.ti.informatik.uni-kiel.de/~kuesters/publications_html/Tuengerthal-IFI-TR-0609-2006.ps.gz.